The saying goes "untested code is broken code".

Parallel-NetCDF has a hodge-podge of tests collected over the years. How well do these tests cover the code base? Let's find out just how un-tested parts of Parallel-NetCDF might be.

Right now I (robl) am exploring what's out there.

- the MPICH2 code has a bunch of perl scripts to collect coverage data and display on a web page the lines that are not covered. Their approach also allows for marking up the code to exclude error handling
- lcov (http://ltp.sourceforge.net/coverage/lcov.php) looks promising, if not really maintained any longer: generate html reports
- a person at sandia has made gcovr: https://software.sandia.gov/trac/fast/wiki/gcovr for a more text-oriented approach to reporting

Preliminaries

You will need coverage information before you can begin to worry about how to present it. r1021 (should be part of the 1.3.0 release) has a '--enable-coverage' configure flag.

```
 ./configure --enable-coverage --with-mpi=/path/to/mpi --prefix=/whatever
```

Then, compile and build as you normally would. Run 'make testing' and any other tests you can think of. The tests will deposit a bunch of stuff.gcda and thing.gcno files in the build directory.

Displaying profile information with lcov

Lcov is a funny tool in that it was developed to show coverage of the linux kernel. So, half of the documentation is about how to display coverage of kernel code, and the other half is for applications. Just something to bear in mind while reading the documentation. Quick summary:

First, turn the big pile of coverage data (the .gcda and .gcno files) into what lcov calls a "tracefile":

```
lcov --directory . --capture --output-file pnetcdf.info
```

This will descend into the directory tree and pick up all the assorted coverage files deposited therein, and put all the coverage information into a single .info file.

Next, make a pretty html report:

```
 genhtml --legend --output-directory pnetcdf_coverage pnetcdf.info
```

Open up pnetcdf_coverage/index.html in a web browser and click away.

Displaying profile information with gcovr

gcovr is a python script. Just run it out of the build directory. Produces a lot of output, but does tell me that for example our tests

> execute 40217 out of 51862 lines, or 77% coverage.